National Défense
Defence nationale

MDR/OMNI-BAND RECONFIGURABLE TERMINAL DESIGN CONCEPT (U)

by

Robin Addison

DEFENCE RESEARCH ESTABLISHMENT OTTAWA
TECHNICAL NOTE 98-007

Canada

September 1998
Ottawa

National Defence / Défense nationale

MDR/OMNI-BAND RECONFIGURABLE TERMINAL DESIGN CONCEPT (U)

by

Robin Addison
Milsatcom Group
Space System & Technology Section

DEFENCE RESEARCH ESTABLISHMENT OTTAWA
TECHNICAL NOTE 98-007

Project
5CA11

September 1998
Ottawa

Abstract

In 1997, Defence Research Establishment Ottawa and Communications Research Centre started research into an omni-band reconfigurable terminal. Such a terminal will enable armed forces personnel to use a single ground terminal to communicate over any satellite communications or terrestrial link. Each ground terminal will support multiple standards. The first one to be implemented will be extremely high frequency medium data rate communications. There are two simulators: the payload simulator and the ground terminal simulator. Each simulator has a personal computer based host to run the simulator, a digital chassis containing several digital signal processor boards to run the payload or ground terminal, and a radio frequency chassis to interface to the digital boards.

Within the simulators, the concentration is on the capability to reconfigure. This will allow easy implementation of different standards for different communications systems. Debugging capabilities will be incorporated at the beginning of the design phase to facilitate development and simplify testing.

The sequence of development, from the initial concept of a medium data rate simulator to industrial development of an omni-band reconfigurable terminal, is given. Initial development concentrates on baseband processing with the radio frequency research in parallel. Midway through, the simulator will be ready for international interoperability testing. Later, additional standards will be supported. Throughout the development, industry will be consulted for the eventual transfer of the technology to industry for development of omni-band reconfigurable terminals for the Canadian Forces.

Résumé

En 1997, le Centre de recherche pour la défense Ottawa et le Centre de recherche des communications ont entrepris une recherche sur une station terrestre reconfigurable à omni-bande. Cette station terrestre serrait capable de communiquer sur n'importe quelle bande, qu'elle soit terrestre ou par satellite. Chaque station supportera une multitude de normes de communication et la première qui sera implantée sera celle de la bande millimétrique à taux moyen de données. La réalisation consiste en un simulateur de satellite et un simulateur de station terrestre. Dans chaque simulateur, il y aura un ordinateur PC pour le commander, plusieurs cartes à traitement numériques et un châssis fréquence radio pour l'interface des cartes numériques.

Il est essentiel que les simulateurs soient reconfigurables. Cela simplifie l'implantation de multiples standards pour les différents systèmes de communications. Dès le début de la conception, la capacité de mettre au point les problèmes sera incorporée pour aider au développement.

Les étapes du développement, commençant par le concept d'un simulateur à taux moyen de données jusqu'au développement industriel d'une station terrestre reconfigurable à omni-bande, sont données. Premièrement, le traitement bande de base sera développé parallèlement à la recherche de fréquence radio. Les essaies d'interopérabilité internationales pourront se produire dès que le simulateur est complété. Les étapes suivantes incorporent les autres standards de communications. Pendant le développement, l'industrie sera consultée pour le transfère de la technologie à l'industrie pour le développement d'une station terrestre reconfigurable à omni-bande pour les forces canadiennes.

Executive Summary

In 1997, Defence Research Establishment (DREO) and Communications Research Centre (CRC) started research into an omni-band reconfigurable terminal. Such a ground terminal will enable armed forces personnel to use a single ground terminal to communicate over any satellite communications or terrestrial link (this is similar to a software radio). On ships, several of these terminals could replace the current multiplicity of communications installations and provide redundancy along with concurrent communications over different links. Each ground terminal will support multiple standards. The first standard to be implemented will be extremely high frequency (EHF) medium data rate (MDR) communications.

The MDR system consists of two main simulators: the payload simulator and the ground terminal simulator. Both will be operated independently, so there is no overall system controller. The simulators will be connected by waveguide or coaxial cable to pass the RF signal between them. Each of the simulators will use similar hardware and will be distinguished by the communication standard (referred to as a personality) loaded in on power-up.

The simulators consist of a host processor, digital signal processing elements and the analog interface. The host processor will handle the user interface, simulator control, event logging, configuration control and download control. The digital signal processing elements will consist of ground terminal or payload controller, access or resource controller, synchronization controller and communications controller. The analog and radio frequency interface to the transmit and receive subsystems is still being researched and will be specified in more detail in subsequent documents.

Within the simulators, the concentration is on the capability to reconfigure using digital signal processors and, if necessary, reconfigurable logic boards. This will allow easy implementation of different standards for different communications systems. The digital signal processor chosen for the design is a powerful state-of-the-art chip from Texas Instruments. Reconfigurable logic boards may also be used to improve real-time performance. Personality information will reside in text files wherever possible allowing modification and customization using a normal text editor.

Debugging capability will be incorporated at the beginning of the design phase. This will facilitate development and simplify testing. It will also ensure a more reliable system as problems will be found early in development. This debugging capability includes built-in test circuits, self tests, sanity checks, bounds checking, as well as the reporting and response mechanisms. The user interface will be able to control the level of debugging and the resultant actions.

The sequence of development is described from the concept of an MDR simulator to industrial development of an omni-band reconfigurable terminal. The first two steps concentrate on baseband processing because the technology is currently more advanced in this area. This processing involves the data communications, synchronization and access control portions. Meanwhile, research is progressing on the radio frequency portion. By the third step, the MDR simulator will be complete and capable of international interoperability testing. The next steps

add low data rate (LDR) and Universal Modem (UM) personalities that allow further interoperability testing. Throughout the development, industry will be canvassed for involvement. The amount and timing of the involvement will depend upon the form of the cooperative venture. The industrial strategy is to cooperatively develop ground terminal technologies and to transfer this technology to industry to support possible Canadian Forces terminal acquisitions.

Table of Contents

Table of Figures

List of Abbreviations

AC	Alternating current
A/D	Analog-to-digital
BER	Bit-error rate
BITE	Built-in test equipment
C6201	Texas Instruments fixed-point TMS 320C6201 digital signal processor
CDROM	Compact Disk Read-only Memory
CMSC	Canadian Military Satellite Communication (project)
CPU	Central processing unit
CRC	Communications Research Centre *also* Cyclic redundancy check
DC	Direct current
DREO	Defence Research Establishment Ottawa
DSP	Digital signal processing
EHF	Extremely high frequency
FSK	Frequency-shift keying
GPS	Global positioning system
GT	Ground terminal
HDR	High data rate (greater than 2 Mb/s)
I	In-phase
IF	Intermediate frequency
I/O	Input/output
LAN	Local area network
LDR	Low data rate (up to 2.4 kb/s)
LNA	Low noise amplifier
LO	Local oscillator
MDR	Medium data rate (4.8 kb/s to 2 Mb/s)
NT	Windows NT operating system
PC	Personal computer
PCI	A high-speed bus used in modern personal computers
PSK	Phase-shift keying
Q	Quadrature (90° phase difference from I)
RAM	Random access memory
RF	Radio frequency
RTOS	Real-time operating system
SDLS	Satellite data link standard
sync	Synchronization
TI	Texas Instruments
TMS320C62x	Family of Texas Instruments digital signal processors
TRANSEC	Transmission security
UM	Universal Modem
VME	A bus used for plug-in processors
x86	The family of personal computer processors including 486 and Pentium

1. Introduction

1.1 Background

In 1997, Defence Research Establishment (DREO) and Communications Research Centre (CRC) started research into an omni-band reconfigurable terminal. Such a ground terminal (GT) will enable armed forces personnel to use a single GT to communicate over any satellite communications or terrestrial link (this is similar to a software radio). On ships, several of these terminals could replace the current multiplicity of communications installations and provide redundancy along with concurrent communications over different links. Each GT will support multiple standards. The first standard to be implemented will be extremely high frequency (EHF) medium data rate (MDR) communications [1]. Capabilities and knowledge developed in this activity will be available to support programs such as the Canadian Military Satellite Communications (CMSC) project.

Future systems could be easily added by supporting a different communications standard (referred to as a personality) for the GT. Two examples are the EHF low data rate (LDR) standard [2] and Universal Modem (UM) standard. There is potential reduction in space and power requirements for vessels as well as reduced training time for operations due to standardization inherent in a reconfigurable terminal. There is also reduced logistical burden and flexibility for commanders to use the most effective means of communications.

1.2 MDR Research

The first activity in the project is to design and build an EHF MDR system including both a payload and GT simulator. Because baseband technology is more mature, the work will first concentrate on the baseband portion. In parallel, more basic research will be required for the radio frequency (RF) component (especially as it relates to omni-band applications). Once this research has indicated the way ahead, the RF portion of the simulators will be developed. The intention is to complete the RF portion at roughly the same time as the baseband portion. The purpose of this research is to develop expertise in digital signal processing (DSP) and RF processing. As well, the system will provide simulators to test concepts, test interoperability with allies and develop an intimate knowledge of the MDR waveform.

The system simulator consists of two main simulators: the payload simulator and the GT simulator. Both will be operated independently, so there is no overall system controller. The simulators will be connected by waveguide or coaxial cable to pass the RF signal between them. The system is described in more detail in the main text.

1.3 MDR Overview

The EHF MDR data link standard is provided in [1]. This is a standard for a processing satellite with extensive robustness inherent to the waveform. The satellite will be a switchboard in the sky. The standard provides for frequency hopping to counteract jammers and permutation to mitigate the effects of smart jammers. In addition, there are various modulation modes to support fixed and disadvantaged terminals. To reduce errors, coding of the data is used. The

complexity and processing requirements of the EHF MDR waveform contributed to its selection as the first waveform to implement.

1.4 Evolution of Detail

The aim of this document is to provide an overview of the concept and detail of the design for the MDR hardware and software. This document, along with subsequent design documents, will contain as much detail as necessary for the construction of simulators. Initially the details will be to the level of detail known and consequently different areas will receive different level of details. As the work progresses and more details are known the design document collection will evolve to reflect this new information. Researchers in specific areas will provide the detail for documentation as it becomes available.

1.5 Outline

This document details the system including GT and payload simulators. Each of the simulators is divided into controllers and processors. The controllers and processors are described along with their requirements and design issues. The controllers and processors are detailed into tasks, data bases and major communications flows. Global issues relevant to most of the blocks are then covered. Finally the planned sequence of development is given.

2. System

2.1 Description

Fig. 1 shows the block diagram for the MDR system simulator. There are two main simulators within the system: the payload simulator and the GT simulator (it should be noted that there is no system controller). The similarity between the two simulators is not coincidental, but rather intentional. Ideally, the simulators should be identical but configured at run-time as payload or GT. Each simulator consists of a personal computer (PC) based host, a digital chassis and an RF chassis.

The host provides operator control, permanent storage, and development capability. This is the only portion that interacts with the user. The digital chassis will contain several DSP boards, potentially some custom general purpose boards, and specialized analog-to-digital (A/D) and input/output (I/O) boards. The digital chassis will address the functionality involved in GT or payload control, access or resource control, synchronization control and communications processing. The RF chassis will include hopping synthesizers, up and down conversion chains with associated amplifiers, mixers and filters. The RF chassis will be based on a modular design to allow substitution for different bands. The digital chassis and RF chassis are covered in more detail in subsequent paragraphs.

2.2 Architecture

The simulators will be controlled by a single x86 host (likely Pentium II or higher). This host will consist of a screen, keyboard, mouse for human interaction. A hard disk and the motherboard will be contained in a VME (or PCI) chassis. The host will control several multiprocessor DSP boards in this same chassis using the backplane for downloading. Input and output of digital or analog data will be through daughter boards on the appropriate DSP board.

It will be necessary to have some custom boards to obtain the required performance. Where possible, these boards should be as general purpose as possible exploiting reconfigurable logic (such as Xilinx), memory for buffering, special purpose chips for coding and other functions. Ideally, there should be only one board design that is used for several different tasks.

A real-time operating system (RTOS) will not be used, instead the DSPs will be programmed directly. An RTOS allows more flexibility (for instance dynamic allocation of task to processors is permitted) but is harder to debug. Direct programming was chosen because it allows rapid debugging and isolation of tasks.

2.3 Differences Between Payload and Ground Terminal Simulators

The payload simulator shown in Fig. 1 is similar to the GT simulator. While the host main program is the same, the DSP processing when configured as a payload is different. The payload must handle several accesses or channels at once and is therefore a more difficult task. This is reflected in a substantially different design between the payload resource controller and the GT access controller and also in the two communications controllers. The synchronization controller of the payload is, however, simpler than that of the GT. In the payload, probes are

detected and estimated and the associated response sent whereas in the GT, the synchronization controller must manage spatial, frequency and temporal acquisition and tracking.

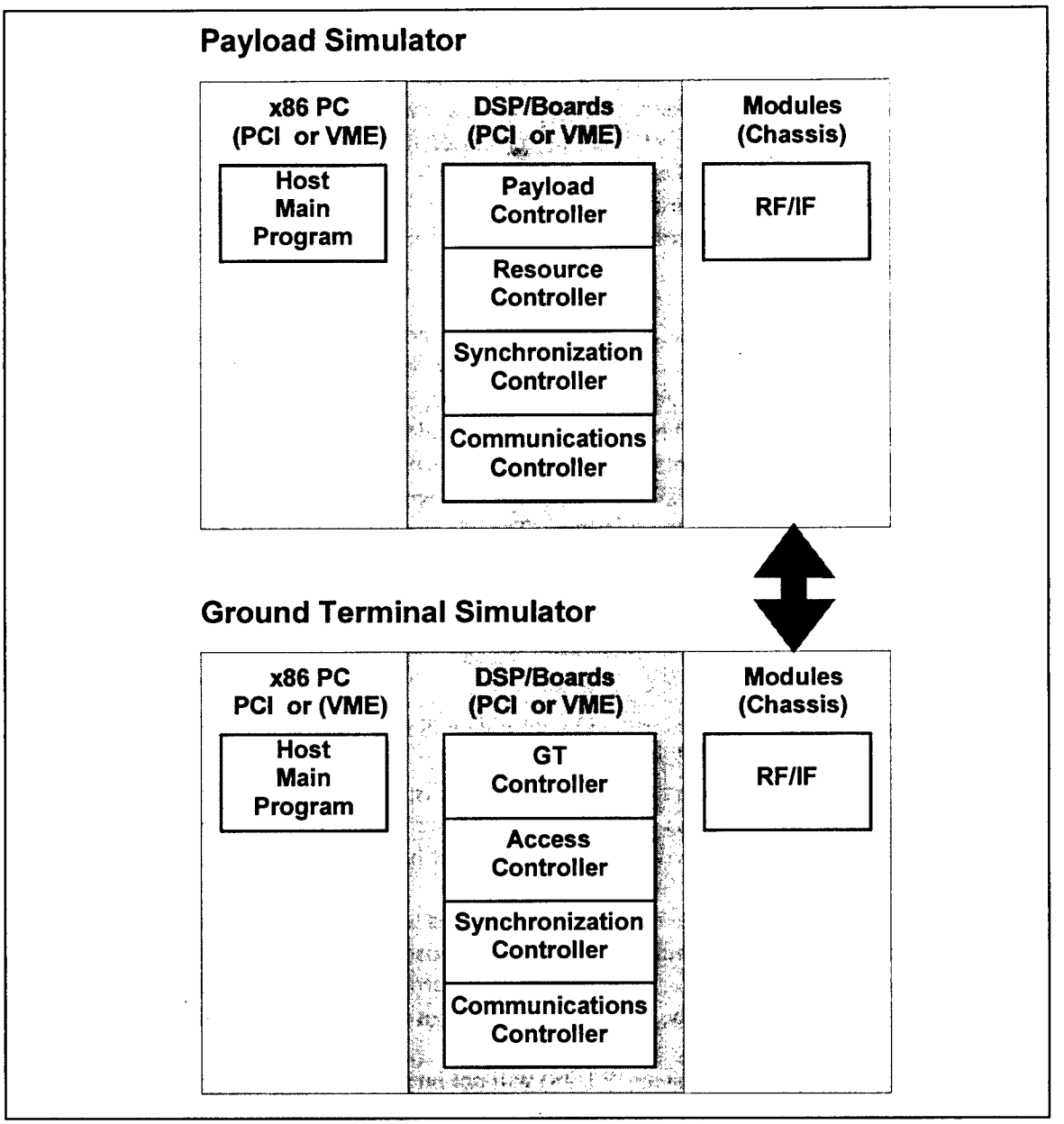

Fig. 1. MDR system simulator block diagram.

3. Host

3.1 General

The host is an x86 PC computer running under Windows NT. The primary purpose for the host within the simulator is to interface with the user to configure and monitor the simulator. Fig. 2 shows a block diagram of the host main program. Major tasks are the user interface, download manager, configuration control and event logger that are detailed in later paragraphs. The BITE(built-in test equipment)/Debugger detail can be found separately in the Global Issues paragraphs. The entire simulator is coordinated via the host simulator control task.

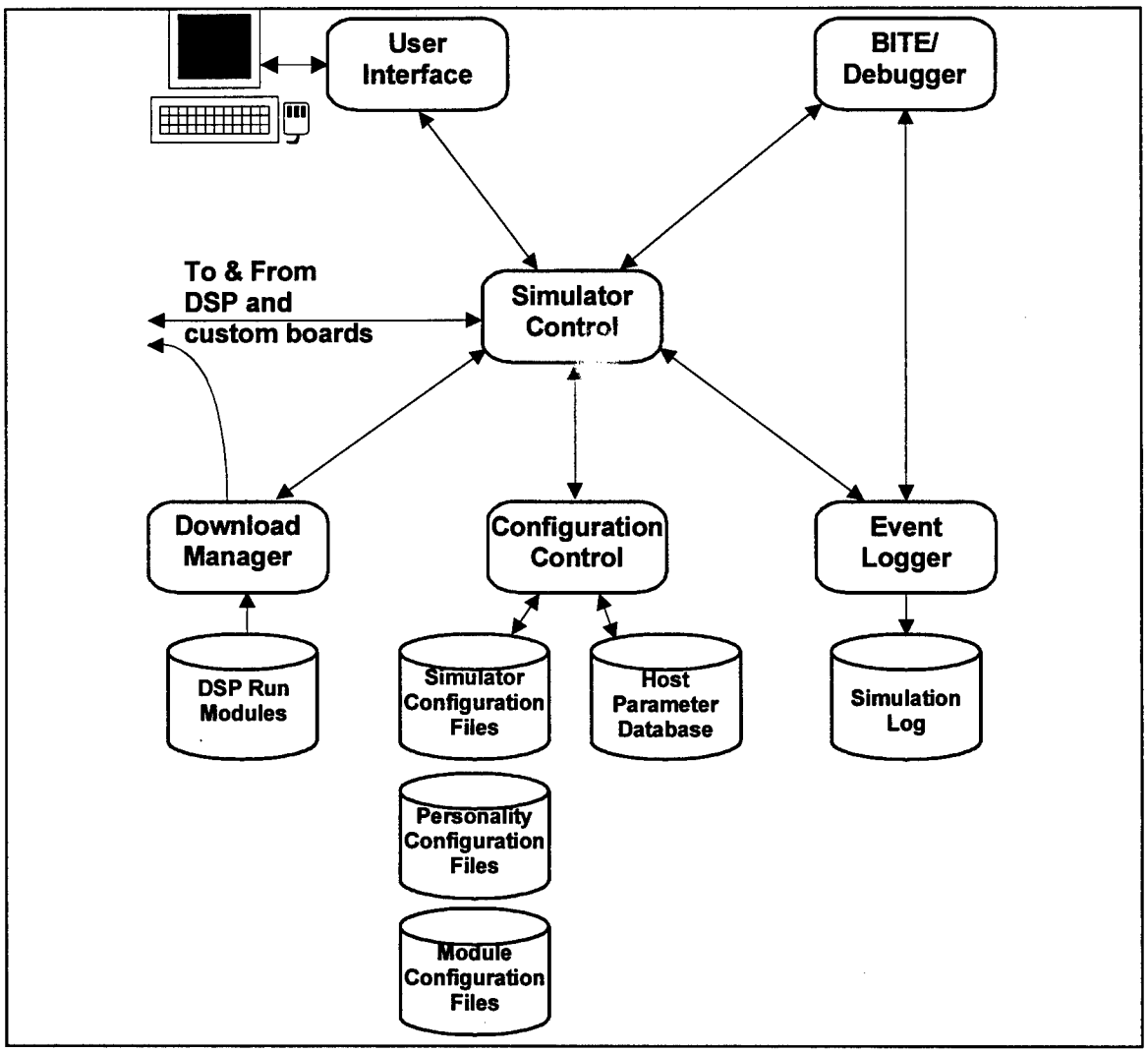

Fig. 2. Host main program block diagram.

3.2 Detail Descriptions

Fig. 2 shows the components of the host main program. Tasks are shown in rounded rectangles. Storage entities (either files or databases) are shown using the symbol for a disk. The key user interface tools (keyboard, display and mouse) are shown as icons. A short description of each component follows:

User Interface

This is the task that transfers human interaction into appropriate commands or directives for the simulator control. It also displays status or information from the simulator control to the user. The user interface task is given in more detail in section 3.3.

BITE/Debugger

This asynchronous task monitors the health and status of the host. It also provides a mechanism for returning debugging information and reporting error conditions. Every part of the simulator has a BITE/Debugger component, the general description of which is given in Global Issues.

Simulator control

This is the overall controller for the simulator. It coordinates the passing of information between components as well as communicating with the DSP and custom boards. It interacts with the user only through the user interface.

Download Manager

This asynchronous task is responsible for ensuring the necessary DSP run modules (executable code and associated data areas) are loaded on the DSPs. The host is the only portion of the simulator with storage. This task is given in more detail in section 3.4.

Configuration control

This task is responsible for loading, saving and managing configuration necessary for the simulator. Different types of files used for configuration are: simulator, personality and run modules. The simulator files are needed to provide the initialization of the simulator itself. Different standards require different initialization that will be provided by the personality files. The run modules provide the DSP executable code to effect the processing necessary for each personality. The simulator information is loaded first, then personality information for the appropriate band/system of operation. Run modules may have special dedicated configuration information that is loaded if required. All configuration information is stored in the host parameter database at runtime and selected parts are passed to the DSPs. This task is given in more detail in section 3.5.

Event Logger

This asynchronous task is responsible for recording and reporting events that occur. The most significant events are errors, but other events such as state changes, status or reports will also be recorded. This task is given in more detail in section 3.6.

3.3 User Interface

3.3.1 Description

The user interface resides on the host and is responsible for transferring human interaction (mouse clicks or key presses) into appropriate commands or directives for the simulator control. The simulator control will then choose the appropriate action and delegate actions to other tasks. The user interface also includes a display to show status or information from the simulator to the user. It should be noted that the partitioning is such that the user interface does not effect changes nor control other tasks (this is the purview of the simulator control task).

Commands could include configuration or reconfiguration, start or stop simulations, request status or report information, override key operations of the simulation (such as force a specific channel to be used for an access). Information to be displayed is the state of the simulator, the selected configuration, the progress of the simulation, event log (filtered or not) and results of a simulation.

3.3.2 Requirements

The user interface must transfer keyboard and mouse actions into host directives or messages. It must also transfer information, status and reporting into a visual medium. Similarly, any information must be printable.

The functionality required includes: configuration of the simulator (including load and save), selection of simulator personality (MDR, LDR, GT, payload), configuration of personality parameters (including load and save), selection of communications (frequency-shift keying (FSK), channel 2, coding), and configuration of communications parameters (including load and save).

The user interface must be able to start, stop, select, display, and clear event log. The error classification and consequent actions must be modifiable at the user interface. While running, the user must be able to monitor simulation progress (state), monitor simulation results, modify communications parameters as well as stop, start and restart the simulation.

For purposes of debugging, the user must be able to view and record the debugging information and show version number or data of any run modules. The user must be able to reload any run modules while running if there is a problem. The user must also be able to override and access or resource information or any state of the simulator.

Miscellaneous functions include the capability to edit and cause to be reloaded any text file, edit/compile/link any run module.

The user interface will be a separate task (or in Microsoft terms - thread) on the host processor so that it is still responsive during downloading or processing of intensive procedures.

3.3.3 Design Points

The user interface should be able to function completely using only a keyboard, a mouse and a display. The user interface should be designed to be intuitive rather than for coding efficiency. It should facilitate information gathering and manipulation of the simulator.

It may be desirable for the user interface to control the power-up sequence of subsystems within the simulator. It should enable individual modules to be turned on or off from the console.

On power-up the user interface should be active as soon as possible. To ensure the user interface is ready, it is important that the operating system boot up quickly without user interaction. The design goal is to achieve less than two minute startup time from power down to running a simulation.

3.4 Download Manager

3.4.1 Description

Unlike the DSPs, the host has hard disk storage and thus, holds all the DSP run modules. The download manager ensures that the necessary DSP run modules (executable code and associated data) are loaded on the DSPs. Downloading takes a long time and it is important that simulator setup is not held up waiting for the downloading to be completed. For this reason, it is vital that the download manager operate asynchronously. The download manager must download run modules at power up and upon personality/communications configuration. It must also download run modules in the event that a configuration change requires new run modules.

3.4.2 Requirements

On power-up, the download manager must download the DSP kernels followed by the default personality and communications software. When the user selects or modifies the personality or communications settings, the download manager must update the DSP software downloads to ensure the required modules are available.

The download manager should be a separate task (thread) to ensure that the user interface is not held up waiting while downloading.

3.4.3 Design Points

In order to reduce download time for the many DSPs in the simulator, it may be necessary to concurrently download several DSPs.

It would be advantageous for downloading to proceed while the simulation is ongoing. This would allow the simulator to switch seamlessly between various modes of operations or personalities.

The download manager may need to read special parameter files in support of downloaded modules. For example, an access control module might require the access message bit field file to be downloaded before running. Some modules might require other (normally independent) modules for various functions. There must be a mechanism to show that a module requires additional files or modules at download time.

3.5 Configuration Control

3.5.1 Description

This task loads, saves and manages configuration information necessary for the simulator. The information is loaded from text files and stored in the host database during running. (There is also a separate subset of this database held by the GT or payload controller on a DSP that can be rapidly disseminated to the other DSPs). Configuration information includes the basic simulator configuration (ex: last used files, GT or payload), personality configuration (ex: MDR or LDR, necessary run modules), communications configuration (ex: data rate, coding, channel, modulation, synchronization parameters) and possibly run module configuration (ex: parameters for modules). This task is also responsible for responding to parameter information requests (possibly allowing for special formats or labels for enumeration types) from all other tasks.

3.5.2 Requirements

The configuration controller must provide, to the host, the ability to load and store the data base, read or modify the parameters and possibly provide secondary information such as meaningful labels for enumerated types.

All parameters will be stored in a data base. The parameters will be saved and loaded from text files with adequate structure and comments to enable changes to be made easily using a normal text editor (such as *Notepad*). There should be the capability of storing multiple files for different setups and the ability to put configuration files in different sub-directories.

All changes to parameters will be made by first updating the database and then referring to the database for reinitialization. Changes can occur when the simulator is running or stopped.

3.5.3 Design Points

The types of text files for parameters are: common initialization file, simulator (GT or payload) initialization file, personality (LDR or MDR) initialization file, and communications setup files (may be multiple per simulation run). There may also be special initialization files for individual modules.

It will be more efficient if the database is hierarchical, possibly using objects. It must still use text files to allow easy editing.

Enumerated types (integer types in C that are referred to using labels rather than values) should be supported by at least defines in a ".h" file and labels managed in the database. All types should probably be supported by formatted string functions for displaying, printing and

logging. There should also be routines to read and write parameters by name rather than some cryptic number.

3.6 Event Logger

3.6.1 Description

This task handles the recording and reporting of all events. Events include various levels of errors, status and state changes of the processors, status of the simulator, results of simulations and possibly debugging information. These events will be stored on the hard disk as the simulation log. The user interface (through the simulator control) will display the log (or a filtered version of it) upon request. There will also be a method of printing the log or a filtered version of it (while the simulator is running). The event logger will also be responsible for filtering the error messages and notifying the simulator control when a serious error occurs (to abort the simulation or possibly shut down).

3.6.2 Requirements

The event logger, in response to an error condition, must be able to: effect power shut-down, abort the simulation, allow operator selection of error action, provide status to the user and provide the error log to the user.

The event logger needs to have event (and more importantly error) management. This involves logging, consolidated reporting, enumeration and parameter decoding, and allocation of errors to subsystems. There must be different error filters for recording, displaying and printing of the log. The errors should be reclassifiable at runtime. This will allow the user to disable or change the class of an error event (and consequently change the resultant action). The error filter profiles should be stored in a text file if desired. There may also be an additional file used during debugging that has temporary overrides of the main text files.

3.6.3 Design Points

The levels of error events that can occur include: code problems (failed error check, index out of range, case out of range, illegal parameter), power shut down errors, simulation abort errors, serious errors (allow the user to specify whether to continue or abort), warnings and information (such as state changes).

Errors must be supported by more than error numbers. There should be a way to obtain a textual error message (including relevant parameters) and a way to refer to the error using defines in the ".h" file. Error message strings and formats (how it is decoded, presented and which parameters are used) should be read in from a text file.

4. Digital Signal Processing

4.1 General

This section details the digital signal processing functions of the GT or payload simulator. These functions will first be implemented on DSP boards. Later, if the DSPs are unable to meet the performance requirements, key functionality will be transferred to reconfigurable or custom boards. Fig. 1, presented earlier, shows the key controllers involved in the DSP portion of the payload and GT simulators. The controllers involved are the GT/payload controller, the access/resource controller, the synchronization controller and the communications controller. Each controller is examined in detail along with requirements and design points.

4.2 GT/Payload Controller

4.2.1 Description

Fig. 3 shows the block diagram of the GT/payload controller. This controller is responsible for control of all DSPs, custom boards and RF/IF (intermediate frequency). It is distinct from the simulator control (which resides in host) because it is running on the DSPs. It also has direct hardware connections to the other DSPs and boards. Aside from controlling the payload or GT there are two miscellaneous tasks assigned to this controller: timing control and TRANSEC control. The controller also has a parameter database that provides information (such as the communications parameters) to the other DSPs. The tasks involved in this controller are detailed below:

Timing Control

This task includes control of the timing information required by every module, DSPs and board. It includes hop clocks, data clocks, and frame clocks. In the GT the clocks are adjusted by the synchronization controller during acquisition and tracking. For MDR, where the satellite has the master clocks, the clock signals are free running in the payload. All clocks, payload and GT, will be derived from the frequency reference that is part of the RF/IF portion.

Transmission Security Control (TRANSEC Control)

This is the task that provides the handshaking and manipulations necessary to get TRANSEC data. TRANSEC data is typically a bit stream used to provide security for the transmission of data such as to control frequency hopping. This data is then passed on to other modules, DSPs and boards to allow for TRANSEC manipulations including frequency hopping, permutation, and cover. This task can generate TRANSEC data from a number of possible sources: externally from attached TRANSEC modules, internally using embedded TRANSEC chips, or internally using special test patterns that aid in debugging.

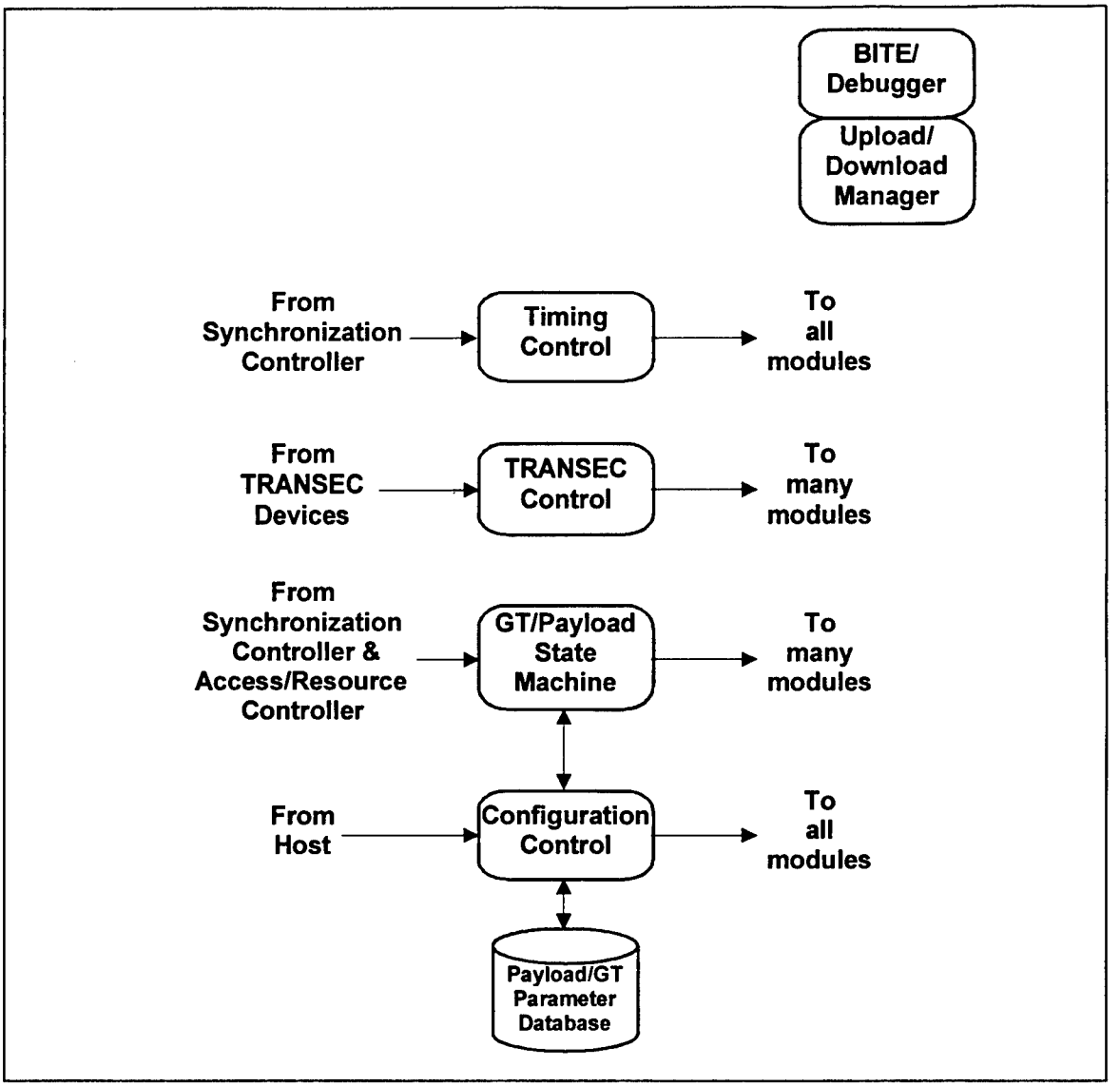

Fig. 3. GT/payload controller block diagram.

GT/Payload State Machine

This is the overall controller of the payload or GT that enforces the sequence of initialization, synchronization, access determination and communications.

Configuration Control

This task manages the configuration information using a local parameter database that is passed to it from the host. This task supplies parameter information to all other DSPs and boards as required.

BITE/Debugger

This asynchronous task monitors the health and status of the GT/payload controller. It also provides a mechanism for returning debugging information and

reporting error conditions. Every part of the simulator has a BITE/Debugger component, the general description of which is given in Global Issues.

Upload/Download Manager

This task accepts run modules and associated data from the host download manager. The downloads occur first at initialization and can occur later at reconfiguration. In the case of reconfiguration, the downloading must be able to occur while the simulator is running. All DSPs have an upload/download manager component, the general description of which is given in Global Issues.

4.2.2 Requirements

A key design requirement for the GT/payload controller is to be reconfigurable while a simulation is running. It must be possible to change various parameters and still continue the simulation. There must also be a debugging capability that can override any requirement or force any state or transition within the controller or any of the subordinate controllers (any DSP board). The controller must be able to log all state transitions, enable or disable synchronization and communications processing on a block by block basis. To aid in hardware troubleshooting with an oscilloscope, there is a requirement for a hardware pulse on the start of any hop or frame, permuted or otherwise.

4.2.3 Design Points

The GT/payload controller will fully utilize at least one DSP. In addition, it will likely require a custom board for timing control and a custom board for the TRANSEC interface.

The states of the GT/payload controller will include: idle, downlink synchronization acquisition, uplink acquisition (downlink synchronized), logged off (terminal synchronized), requesting logon, no accesses (terminal logged on), requesting access, and communicating.

The timing controller must be settable by an external clock (such as global positioning system (GPS)) or by the operator. Once the time is set, the clock should be started using a trigger. The trigger could be either an external pulse or an operator typing a key.

In addition to interfacing to an external TRANSEC device, the TRANSEC controller will require special TRANSEC patterns (such as the Magic 5 [3]) for debugging purposes. Consideration should be given to a software TRANSEC or an embedded chip TRANSEC if the software or hardware can be obtained from the United States.

4.3 Access/Resource Controller

4.3.1 Description

Fig. 4 shows the composite block diagram for a payload resource controller and a GT access controller. It should be noted that blocks labeled GT will not be found on the payload and blocks labeled payload will not be found on the GT. The payload resource controller is responsible for allocating communications resources to terminals and their users. It keeps track

of available resources, accesses, terminals and satellites in databases. Access request messages are received and decoded and then acted upon generating responding access response messages. The GT access controller is responsible for requesting communications resources on behalf of the terminal and associated users. Access request messages are generated and responses decoded to accomplish this. The access and constellation information are stored in a database. The access/resource controller tasks are detailed below:

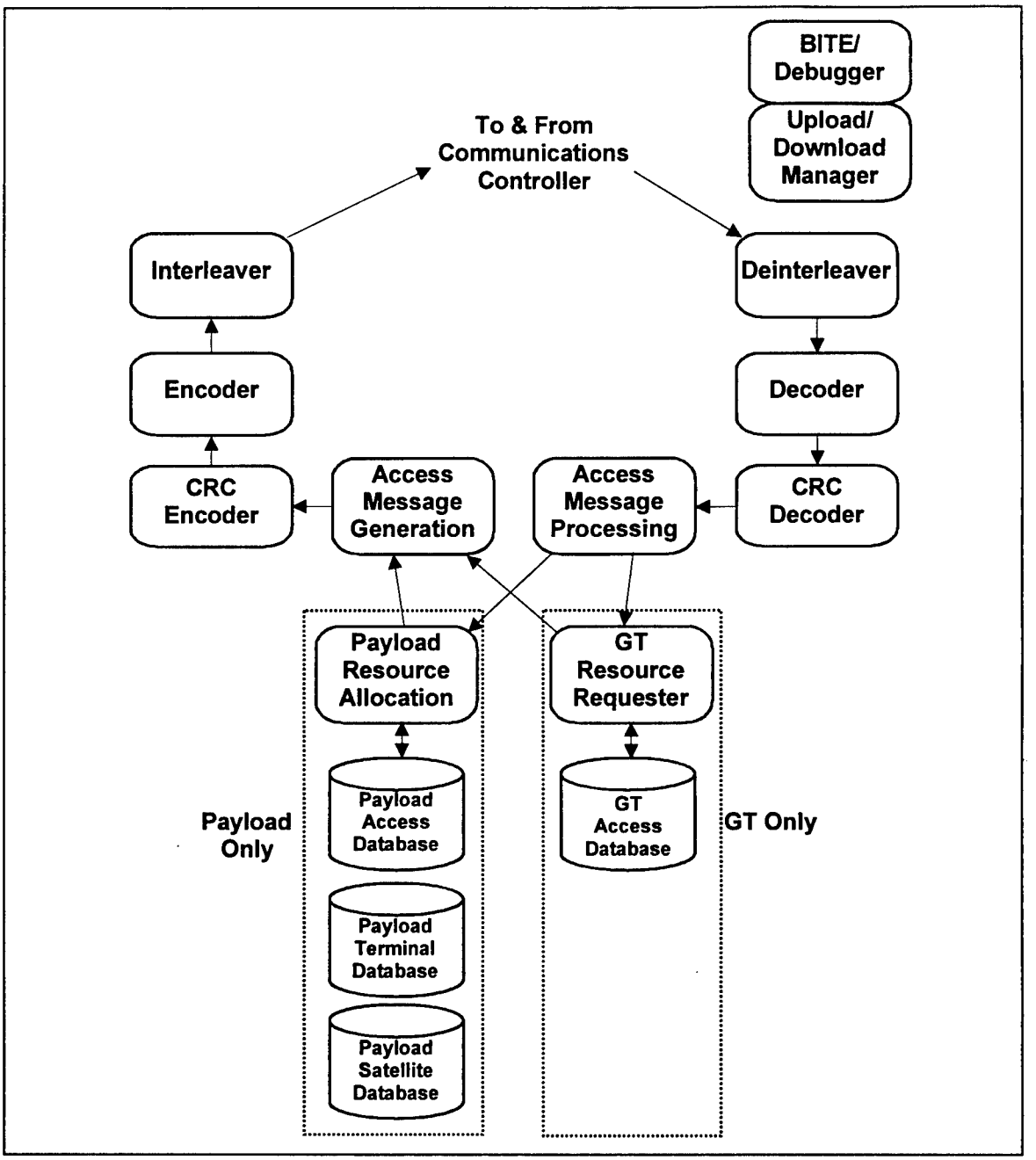

Fig. 4. Access/resource controller block diagram.

Interleaver/Deinterleaver

These tasks are part of the data manipulation to improve the messaging reliability. Access messages are interleaved to distribute the effects of burst errors.

Encoder/Decoder

These tasks are part of the data manipulation to improve the messaging reliability. Access messages are encoded to mitigate the effects of noise and jamming.

CRC Encoder/Decoder

These tasks are part of the data manipulation to improve the messaging reliability. Access messages are cyclic redundancy check (CRC) encoded to validate the message.

Access Message Generation

This task formats the bit fields and generates an outgoing access message in response to a request from the GT resource requester or the payload resource allocation.

Access Message Processing

This task formats extracts the relevant fields from an incoming access message and passes the data to the GT resource requester or the payload resource allocation.

Payload Resource Allocation

This task allocates communications resources in the payload in response to requests from the GTs. Information about the accesses and free resources is stored in the payload access database. Two other databases store information on GTs and satellite/constellations.

GT Resource Requester

This task requests communications resources from the payload in response to requests from the GT users. Information about the allocation is stored in the GT access database.

BITE/Debugger

This asynchronous task monitors the health and status of the access/resource controller. It also provides a mechanism for returning debugging information and reporting error conditions. Every part of the simulator has a BITE/Debugger component, the general description of which is given in Global Issues.

Upload/Download Manager

This task accepts run modules and associated data from the host download manager. The downloads occur first at initialization and can occur later at reconfiguration. In the case of reconfiguration, the downloading must be able to occur while the simulator is running. All DSPs have an upload/download manager component, the general description of which is given in Global Issues.

4.3.2 Requirements

The access/resource controller will be based on state machines. In the payload simulator, the resource controller will have one state machine per access and one state machine per logged on GT. Within the GT, there will be one state machine per user and an additional state machine for the terminal logon state.

To maintain the capability for reconfiguration, it is important that the states, transitions and messages be programmable. Ideally the whole protocol should be loaded from a file when initializing the controller. This requires that the coding of the controller be as general as possible, maximizing the use of defines or parameter files.

For testing and debugging, it is important that the operator can override any requirement for transition, generate any message (including illegal messages), act as if any message is received, control which resources are automatically allocated, or manually allocate any resource. It should also be possible to enable or disable the parity, cover, interleave or coding. State transitions and messages must be passed to the event manager for logging and display. For hardware debugging, it is important that a pulse be generated at the beginning of any message or message cycle.

4.3.3 Design Points

There will be data bases in the payload for GTs and accesses. Additional data bases may be needed for beams and users. The GT will have a data base for accesses, constellations, ephemeris and users.

The access/resource controller will fully utilize at least one DSP. It is unlikely that any custom boards will be required.

4.4 Synchronization Controller

4.4.1 Description

Fig. 5 shows the composite block diagram for a payload and GT synchronization controller. It should be noted that blocks labeled GT will not be found on the payload and blocks labeled payload will not be found on the GT. The payload synchronization controller is responsible for the generation of payload synchronization aids, the detection and estimation of received probes as well as the generation of the associated responses. The GT synchronization controller is responsible for the detection and estimation of the received synchronization aids, the generation of synchronization probes as well as processing of the probe responses. Synchronization primarily concentrates on temporal synchronization but can also include frequency and spatial synchronization. The GT synchronization controller modifies the clocks in the GT controller to effect timing changes. The synchronization controller tasks are detailed below:

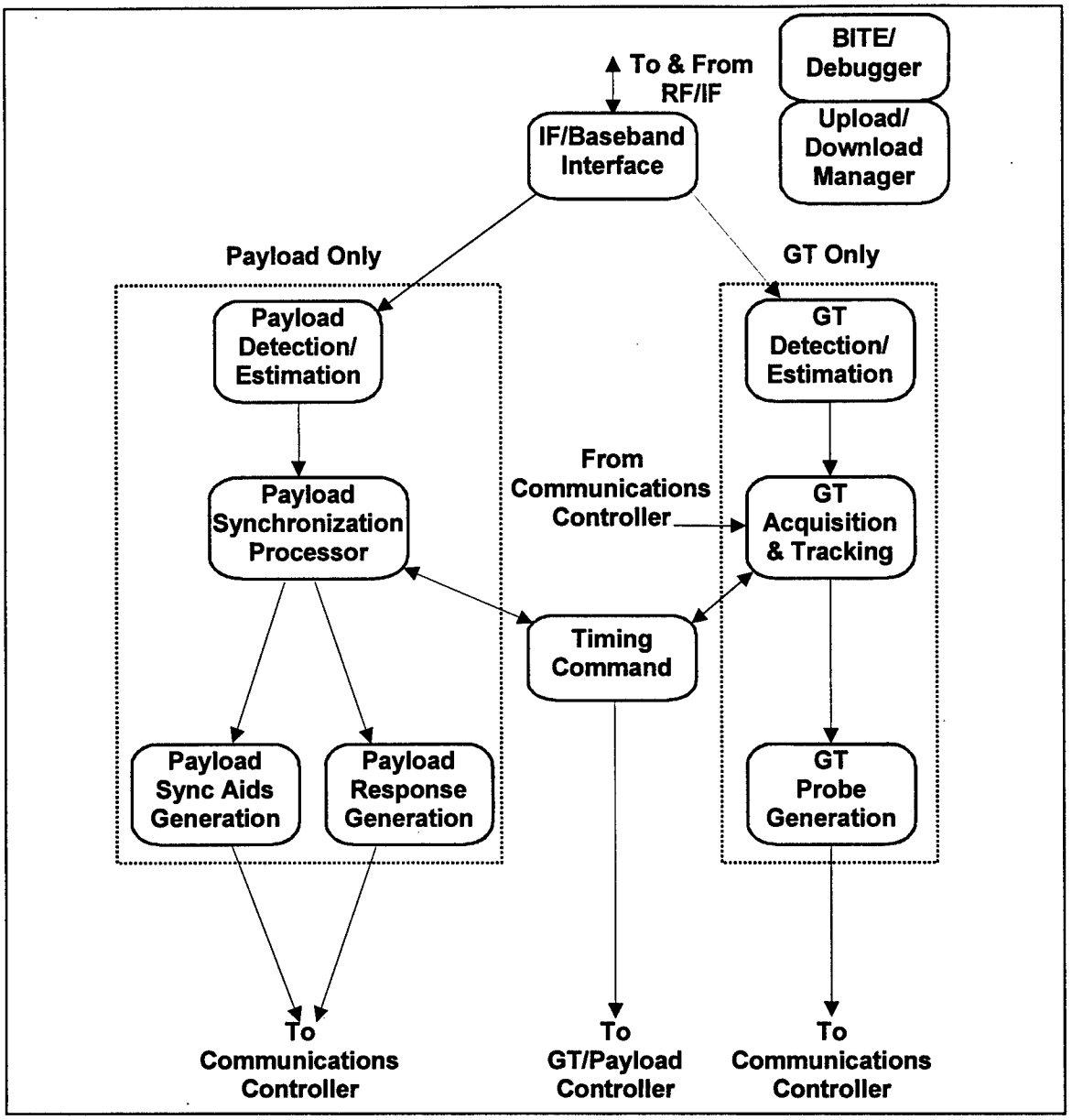

Fig. 5. Synchronization controller block diagram.

IF/Baseband Interface

This task is responsible for conversion of analog signals into digital samples. The interface could be either at an IF or complex baseband. Digital samples are then passed on for detection and estimation.

Payload Detection/Estimation

This task examines the digital samples received to determine a time error estimate or a detection of a probe at the payload.

GT Detection/Estimation

This task examines the digital samples of a synchronization aid received by the GT to determine a time estimate or a detection.

Payload Synchronization Processor

This task takes the payload detection and estimation information and causes the necessary responses to be generated. It also causes the synchronization aids to be generated. While it is not required for the MDR standard, it may be important in other standards and during debugging, for the payload synchronization processor to have the ability to change the clock. The term processor is distinct from controller. (This processor is only part of the payload synchronization controller.)

GT Acquisition & Tracking

This task takes the GT detection and estimation information and changes the clocks to allow the GT to acquire and track the payload clocks. The task also determines the state of synchronization and calculates time offsets. In addition, the task determines when synchronization probes are to be sent to the payload.

Timing Command

This task generates commands to be sent to the GT/payload controller to modify the clocks.

Payload Synchronization Aids Generation

This task generates the synchronization aids (synchronization hops or beacons) from the payload.

Payload Response Generation

This task generates the synchronization responses (detection or estimate) from the payload in response to a probe from the GT.

GT Probe Generation

This task generates the synchronization probes from the GT.

BITE/Debugger

This asynchronous task monitors the health and status of the synchronization controller. It also provides a mechanism for returning debugging information and reporting error conditions. Every part of the simulator has a BITE/Debugger component, the general description of which is given in Global Issues.

Upload/Download Manager

This task accepts run modules and associated data from the host download manager. The downloads occur first at initialization and can occur later at reconfiguration. In the case of reconfiguration, the downloading must be able to occur while the simulator is running. All DSPs have an upload/download manager component, the general description of which is given in Global Issues.

4.4.2 Requirements

The GT synchronization controller must provide the detection and estimation of downlink synchronization hops. It must also generate uplink synchronization probes. This process occurs during synchronization to acquire and during communications to track. During synchronization, the controller must be able to modify the uplink and downlink timing, frequencies and spatial pointing.

The payload synchronization controller must generate downlink synchronization hops and any other synchronization aids (such as a beacon). It must provide the detection and estimation of uplink synchronization probes and then generate the appropriate downlink synchronization response messages. For test purposes (and possibly for other standards), this controller must be able to modify uplink and downlink timing.

4.4.3 Design Points

The synchronization controller will fully utilize at least one DSP. A custom board may be required to provide the necessary performance in processing. It is likely that the synchronization controller will require a separate A/D (distinct from the one used for the demodulator) because of different filtering requirements. However, the design goal is to use only one A/D for demodulation and synchronization.

The synchronization controller will need to have the capability to command the timing controller of the GT/payload controller.

4.5 Communications Controller

4.5.1 Description

Fig. 6 shows the composite block diagram for a payload and GT communications controller. It should be noted that some blocks (that are used only for communications between GTs and not processed in the payload) will not be found on the payload. The GT communications controller takes the user data, performs manipulations to improve reliability and then formats and modulates for transmission. The reverse operations occur for receive. Some blocks (manipulations such as coding and interleaving) will have a separate chain for each access or user. The payload communications controller demodulates all channels, reverses manipulations, extracts the data and then manipulates, formats and modulates for retransmission. The communications controller tasks are detailed below:

IF/Baseband Interface

This task is responsible for conversion of analog signals into digital data and conversion of digital samples or phase/frequency information into an analog waveform. The interface could be either at an IF or complex baseband. It is possible that the transmit modulation could be done at the synthesizer rather than at IF or baseband. In that case, there would be no IF/baseband interface on the transmit side - instead there would be digital controls from the modulator to the synthesizer.

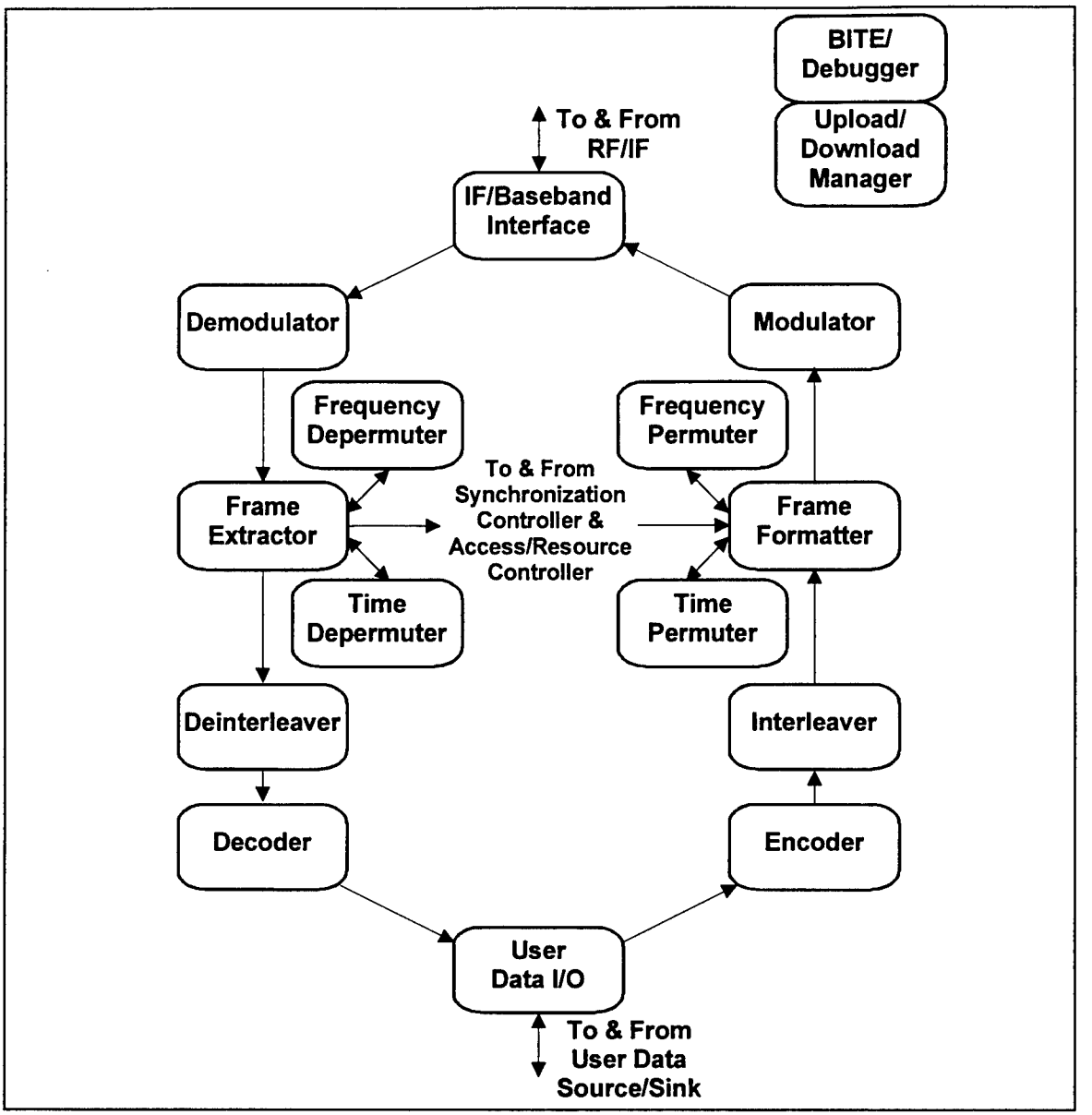

Fig. 6. Communications controller block diagram.

Demodulate/Modulate

This task provides the interface from the digital to the analog waveform. For demodulation, the digital samples of the received baseband waveform are converted into receive bit streams. For modulation, the transmit bit stream is converted into digital samples of the transmit baseband waveform. Alternately, the modulator could produce phase/frequency information that would be sent directly to the frequency synthesizer. In the case of the payload, the demodulator must be able to demodulate several channels or bit streams concurrently. The demodulator must be able to make hard or soft decisions.

Frequency Permute/Depermute

These tasks are part of the waveform manipulation to improve the robustness. The permutation is driven by the TRANSEC bit stream. Data is permuted to mitigate the effects of jamming.

Frame Extractor

This task takes the demodulated symbols or bits and extracts the user, access control and synchronization response bit stream and routes them to the appropriate processor. This task causes depermutation to occur. If symbol diversity is being used, the symbols will be combined here. The frame extractor must be able to handle soft decisions from the demodulator.

Frame Formatter

This task takes the user, access control and synchronization probe bit streams and formats them in sequence for transmission. This task causes permutation to occur. If symbol diversity is being used, the symbols will be repeated within the frame formatter, prior to permutation.

Time Permute/Depermute

These tasks are part of the waveform manipulation to improve the robustness. The permutation is driven by the TRANSEC bit stream. Data is permuted to mitigate the effects of jamming.

Interleaver/Deinterleaver

These tasks are part of the data manipulation to improve the data reliability. Data is interleaved to distribute the effects of burst errors.

Encoder/Decoder

These tasks are part of the data manipulation to improve the data reliability. Data is encoded to mitigate the effects of noise and jamming.

User Data I/O

This task accepts and outputs user data bit streams. In the GT the user data I/O task occurs in hardware (likely a DSP daughter board) and is connected to a data source/sink. In the payload, this task places demodulated data into a buffer that is subsequently retransmitted. Any necessary encryption for classified data will take place outside the simulator. Thus the user data is either unclassified or already encrypted (therefore can be treated as unclassified)

BITE/Debugger

This asynchronous task monitors the health and status of the communications controller and associated boards. It also provides a mechanism for returning debugging information and reporting error conditions. Every part of the simulator has a BITE/Debugger component, the general description of which is given in Global Issues.

Upload/Download Manager

This task accepts run modules and associated data from the host download manager. The downloads occur first at initialization and can occur later at reconfiguration. In the case of reconfiguration, the downloading must be able to occur

while the simulator is running. All DSPs have an upload/download manager component, the general description of which is given in Global Issues.

4.5.2 Requirements

The communications processor must provide modulation and demodulation for both uplink and downlink. Frame formatting and deformatting along with permuting and depermuting are necessary [1]. Each user or access requires separate coding and interleaving and the associated decoding and deinterleaving.

The demodulators must make hard and (when selected) soft decisions that can be carried to at least the frame extractor for diversity combining. The frame formatter will incorporate symbol repetitions when diversity is selected. The frame extractor must be able to function during the synchronization process as well as during normal communications.

4.5.3 Design Points

The frame formatter/extractor should control the hopping synthesizer with respect to the hop frequency. If modulation is done at the synthesizer, then the modulator must also have control of the hopping synthesizer - at least the phase control for phase-shift keying (PSK) and fine frequency control for FSK.

Alternately, diversity generation and combining could be incorporated in a separate module from the frame formatter and extractor. Detailed investigation of the waveform may indicate that it is necessary to incorporate diversity processing into the modulator and demodulator. Further research is necessary before selecting the most effective method.

The communications controller will initially fully utilize at least one DSP board. Later it is likely that processing performance will require off-loading of some of the modules (such as coding or interleaving) to reconfigurable logic boards possibly with special purpose coding chips.

5. RF/IF

5.1 Description

The RF/IF portion is still being researched and the configuration is yet to be finalized. In general, this portion will start at baseband from the modulators and upconvert with a hopping local oscillator (LO) to a hopped IF and then upconvert to the RF frequency. The downconversion chain is the reverse. The simulators will be linked at RF through waveguide or coaxial cable.

Current research involves investigation of single step up and downconversion with a hopped LO. Fig. 7 shows the block diagram for a single step conversion chain. There is potential for design simplification and easy integration of multiple bands if the single step conversion technique proves to be feasible.

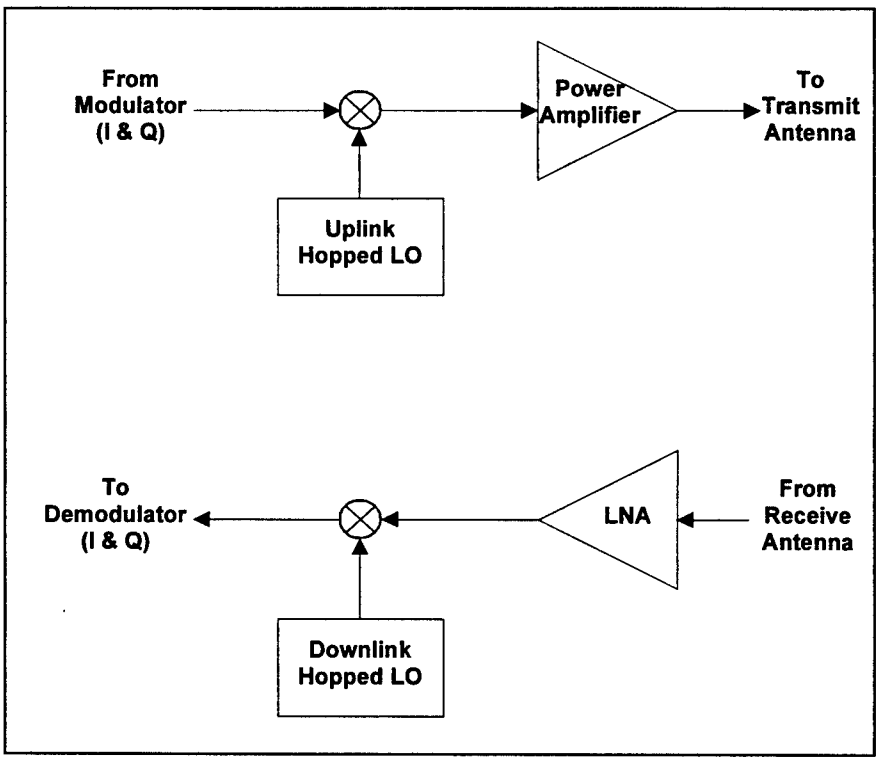

Fig. 7. RF/IF block diagram.

Since a primary goal of the omni-band reconfigurable terminal is to handle many bands, modular construction may be necessary for the RF/IF. Ideally all components in the RF/IF chain would support all bands. However, it is more likely that the antennas, low-noise amplifiers (LNA), power amplifiers and final conversions will be unique for each band. Therefore initial designs should consider a modular approach for these components.

5.2 Requirements

It is important, because of the multiple frequency bands that are to be supported, that a flexible frequency plan be developed. The frequency plan must incorporate a common frequency reference (that can also be supplied externally), hopping synthesizers, RF to IF conversion, hopping and dehopping, IF to baseband conversion, amplifiers, and power supplies. There must also be built-in monitor points and injection points. For testing and debugging purposes, power levels, phase locks and direct current (DC) levels should be monitored.

5.3 Design Points

Current hopping synthesizers have the capability of phase modulation (as well as FSK modulation). Depending on the design, it may be more efficient to modulate the synthesizer rather than to provide baseband modulation. Certain types of modulation have to be implemented at baseband because of filtering and shaping that cannot be done at the synthesizer.

The modular approach for RF/IF design would dictate that each family of frequency bands (bands that are very similar in frequency and can use the same filters) be supported by a single plug-in module for both amplification and conversion. Such a module would have the following inputs: receive signal, frequency hopped LO, alternating current (AC) or DC power, frequency reference, modulation baseband, test injection ports and configuration control. The outputs would be: transmit signal, baseband to demodulator, status and monitor ports.

6. Global Issues

6.1 General

This section details issues which affect all or most processors/controllers/modules. Hardware issues are covered first, followed by software. The portion of the simulator that can be found in almost all modules, for debugging and self-test, is also examined. Finally, common feature of many modules, the upload/download manager, is detailed.

6.2 Hardware

6.2.1 General

The host and all DSP hardware will reside in one chassis. The display and keyboard will necessarily be separate. The chassis will be either of a VME or a PCI form factor that allows plugging in of the host computer, DSP boards, A/D boards, I/O boards and any custom logic boards. RF hardware will be in a separate modular chassis (details are given at the end of section 5).

The storage requirements of the host include at least one large removable hard disk, a CDROM drive for software installation, a backup device (likely a tape drive), and a floppy disk (or possibly a Zip drive) for data transfer between computers. The hard disk must be large enough for the host including the user interface, the host development system, debugging the user interface, the DSP development system, debugging DSPs and multiple download versions.

The host should also be local area network (LAN) capable for inter-simulator communications. It is important that support for the network does not adversely affect the real-time processing capability of the host. The host will also need a printer for hardcopies of logs, reports, configuration, results, and listings.

6.2.2 Use of Boards

Initially, all processing will be performed by the DSP boards. Later, when the time critical portions are identified and it is deemed necessary for performance reasons, the time critical portions may be migrated into reconfigurable logic boards. If reconfigurable logic is still not fast enough then a custom board may be used. Custom boards are to be considered only as a last resort because they reduce the efficiencies in the reconfigurable terminal.

To minimize the costs to the project, the A/D should be ordered as late as possible in the design cycle. This delay will allow for maximization of savings brought about by advances in A/D technologies and performance.

It may be necessary to have the A/D and I/O modules installed directly on the DSP boards to achieve the throughput desired. In this event, consideration should be given to having the designers of the DSP boards develop the A/D and I/O modules.

6.2.3 Reset

While debugging, there will be a requirement to be able to reset the hardware and software at various levels. Essential resets are: complete reset and reset all modules but the host. It is preferable to have reset controllable through the user interface on the host.

It is imperative that all memory devices including random access memory (RAM), flip-flops, shift registers, configuration latches be capable of being reset.

6.2.4 Chassis

There are two options for the choice of chassis: VME or PCI. A VME-based chassis has the advantage of allowing larger sizes of boards (allowing more capability or more DSPs on each board and reducing interconnects between boards) and the advantage of the availability of a large number of boards of diverse capabilities due to the maturity of the technology. The disadvantage of a VME-based chassis is the cost of boards.

A PCI-based chassis, on the other hand, supports the same boards that go into PCs and thus achieves the economies of scale. Many modern A/D and special purpose boards are now being developed only for the PCI bus. The down side of PCI is the size of the boards that reduces the number of DSPs per board and necessitates more interconnects between boards.

6.3 Software

6.3.1 Host

The host will be PC-based and use Windows NT version 4.0 with Service Pack 3. The host software will be developed using Microsoft Visual C++ version 5.0 and using the Microsoft Foundation Class library. Multithreading should be used to ensure the user interface is not hung up waiting for operation to complete. As well, it is important that the user interface start quickly and not be delayed by the downloading operation.

6.3.2 DSP

For the implementation of an omni-band reconfigurable terminal, the DSP software will be developed using the Texas Instruments (TI) C compilers with Spectrum libraries for communications between the host and DSP boards. All software for the DSPs will be initially written in C. Once the software is running, critical code may be identified and optimized. Finally, if necessary for real-time performance, key routines will be coded into assembler language.

6.3.3 Development Practice

To ensure repeatability of software versions, the building of individual modules and the complete build of the simulator will only use 'make' files or project building equivalents.

All software will be controlled by version number. The version will be included as part of the comment and may be passed upwards through a subroutine call. All modules will be

All software will be controlled by version number. The version will be included as part of the comment and may be passed upwards through a subroutine call. All modules will be independently tested (by another team member) prior to a new version being incorporated into the current build. The team leader will be the archiver and control which versions of each module are in the builds.

To maximize flexibility and capability for reconfiguration, all parameters are stored in a database. No parameters are hard coded except for debugging purposes.

Debug versions of modules will be used during the development of the simulators. Only when all modules are complete will non-debug versions be used. The debug versions should perform maximum error checking, and possibly use conditional compilation to later omit the less essential error checking. For the error checking purposes, the following should be observed:

a. all *case* statements need *default*

b. all *case*-like *if* constructs need *else*

c. there must be no infinite loops, all loops must have another exit (time, key press, trigger event)

d. there must always be a way to abort a module on error check failure

6.3.4 Filenames

There must be a consolidated naming system that allows for unique names. The long file names of Windows NT should be exploited to provide meaningful names. It may be preferable, and clearer, to use sub-directories to resolve ambiguity rather than use overly complex names.

6.3.5 Comments

All modules must contain comments including a header, internal comments explaining algorithms and techniques, as well as comments explaining the use of variables. The header must include the name of module, author, hierarchy of module, purpose of module, inputs, outputs (including file or database changes), return values, compilation method (special compiler or flags), date, and change history.

The requirement for comments can be reduced by maximum use of database parameters and defines with descriptive names. Use of parameters will eliminate numbers in the code. As well, extensive and rigorous use of C++ objects in the host can reduce commenting.

6.3.6 Run Modules

For maximum flexibility, all modules should accept various sized blocks of data as input. Details of such data packets can be found in a separate technical note [4]. Input parameters will not be modified. The only parameters that can be changed are output (such as processed data), input/output (such as the run module's state information) and the return value of the call.

Since modules may be used by different parts of the simulators (for example the coder is used by the data communications controller as well as the access/resource controller), all modules must allow multiple independent instances. To allow for multiple instances, all state information required by a module must be supplied by the calling routine.

During setup, run modules must accept initialization. The parameters for initialization must come from the data base. Any special parameter files required by the module must be indicated to the download manager during initialization.

Changes in the simulation may result in a requirement to reinitialize modules or change their parameters. All modules must accept changes or reinitialization while running. Any change to the parameter database should always precede reinitialization. The database should be the only source of parameters for a routine. To ensure the internal modules of the simulator are synchronized, it may be necessary to schedule when reinitialization will occur (such as on the next frame or hop).

6.4 BITE/Debugger

There is a requirement within the modules, both hardware and software, for sections of code or circuits to aid in debugging and to provide the capability for self-test. This capability is beneficial during the development of the simulators both for sanity checks and to isolate problems. This capability will also be used in the running version to ensure the integrity of the simulator.

Hardware error checking must be incorporated into the design from the beginning. All software should have built-in error checking to detect an illegal case or an out-of-bounds index. It may be necessary to disable part of this capability for the running simulator (not debugging) to optimize real-time performance.

Once an error or problem is detected, the information must be passed back to the event manager and then the user interface. This passage of information must be asynchronous to ensure it is not lost. Exceptions may be the best technique to be used to achieve error reporting. Debugging information may be passed a value at a time or as a large data packet. Error reporting must be a high priority task and cannot be held up by loops or other processing.

6.5 Upload/Download Manager

Within each DSP, support is required to the kernel to allow programs and data to be downloaded from the host. Part of downloading may be a read-back function that would require the program or data to be uploaded back to the host. Downloads and uploads should be checked for corruption using error checking.

The upload/download manager residing in the kernel must allow dynamic downloads. This means that the downloads can occur while the simulator is executing. To accomplish dynamic downloads, the manager must be able to: allocate memory, free up unused modules, pass linking information to running tasks and report when a download is complete.

7. Sequence of Development

7.1 General

This section covers the sequence of development or building. It starts with the most fundamental MDR data manipulations and progresses to the complete omni-band reconfigurable terminal. The breakdown is based on achievable and testable results at the end of each step. The early effort concentrates on the most essential portions (and incidentally the simpler portions) of the reconfigurable terminal.

7.2 Step 1 - Data Communications

The purpose of this step is to implement the data manipulation functions that are used for user data communication using the MDR data link standard [1]. The work in this step does not include any of the synchronization or access control portions that would be done prior to a data link being established. Development and testing of the data manipulation functions require simplified implementations of certain areas of the terminal control and user interface. These areas will not be developed significantly in this step, but rather will only be developed to the minimum necessary to test data communications. At the end of this step, the hardware should be able to perform a digital test with a bit-error rate (BER) analyzer including testing with injected errors. This step includes the following items:

a. Design and planning of the reconfigurable omni-band terminal.

b. Learning the new C6201 DSP boards and developing the interfaces and shells necessary to develop and test run modules on these boards.

c. Learning the MDR data link standard to the extent necessary for implementing the data manipulation functions as well as achieving an overall understanding of the MDR waveform.

d. Develop the run modules necessary for all data communications functions. The functions include coding, interleaving, modulation, permutation, formatting, inputting and outputting of the user data. It also includes the functions for data manipulation of access control messaging that are in common with user data manipulation.

e. Develop the minimum necessary for the host processor to be able to develop and run the data manipulation modules. The minimum necessary includes sparse implementations of the simulator controller, user interface, event manager. It also includes complete implementations of the download manager and the parameter database.

f. Research the RF portion of the omni-band terminal. Initial examination of one or two-stage downconversions to baseband in a single device is planned. Future research may include multiple bands using either multiple devices or more complex devices.

7.3 Step 2 - Synchronization

The second step is to include synchronization at both the payload and GT simulators. The GT simulator should now be able to synchronize to the payload simulator without a time reference. Once synchronization is achieved, communications can occur on pre-established channels. The user interface must be developed to enable channels to be selected. A GT/payload controller will also be required to coordinate synchronization and data communications. At this point, some of the custom boards may be developed to improve performance. The items included in this step are:

a. Validation of the architecture. Deficiencies that are identified may result in modification or additions to the planned architecture.

b. Validation of the parameter database. It is expected that throughout the development, parameters will be added to or refined for the database.

c. Develop run modules for all synchronization functions. Synchronization requires special functions such as estimation, clock manipulation and the synchronization controller itself. Also, modifications are required to the data communications to support synchronization aids.

d. Develop run modules for the GT and payload simulator controller. A controller is required to coordinate the resources of the simulator, specifically to initiate synchronization and to only enable communications once synchronization is achieved.

e. Improve the simulator controller within the host to support synchronization.

f. Develop run modules for basic access/resource controllers. This would be the skeleton of the controllers. Only the portion of the access/resource controller necessary to allow synchronization will be implemented.

g. Complete the user interface for a single personality - specifically MDR. All the options and controls need to be installed in the user interface even if the capability has not yet been included in the simulator.

h. Complete the event manager for the simulator. As in the previous paragraph, all the options and controls need to be implemented even if the capability has not yet been included in the simulator.

i. Improve processing for data communications and synchronization functions using optimized or assembly code supported by custom boards.

j. Investigate and then plan RF for MDR band remaining cognisant of the eventual conversion of the RF to omni-band.

7.4 Step 3 - MDR Done

This step includes the completion of the MDR simulator with all the necessary software, hardware and RF for MDR and the hooks in place for other personalities. At this point, interoperability testing may be considered with the United States (likely at Lincoln Lab). The following items are included in this step:

a. Complete the access/resource controller for the MDR personality. This includes full access/resource control as well as the ability to debug, override resource allocation and simulate messages that are used for interoperability testing.

b. Complete the simulator controller to support the access and resource control as well as associated debugging.

c. Develop the RF hardware for MDR. This should be as flexible as possible to allow for minimum changes to support other personalities.

d. If feasible, conduct interoperability tests of the MDR simulator with the simulator at Lincoln Lab.

7.5 Step 4 - LDR Done

The LDR data link standard [2] is very similar to MDR and is logically the next personality to be developed. The hardware is almost identical as well as the frame structure and processing functions. Some additions will be required to fully support multiple personalities. At this point, the research has demonstrated the proof-of-concept of a reconfigurable terminal (though the omni-band portion has not yet been proven). The following activities are included in this step:

a. Improve the simulator controller on the host to support multiple personalities. This will enable switching back and forth between MDR and LDR as well as developing a composite LDR/MDR personality.

b. Improve the user interface to support multiple personalities.

c. Complete the LDR personality. The RF hardware is unchanged and most of the processing the same. There are a few additional types of processing required.

d. The LDR personality could be validated by further interoperability tests at Lincoln Lab.

7.6 Step 5 - Universal Modem Done

Unlike the LDR, which uses the same band and processing as MDR, the Universal Modem is quite distinct (the standard has yet to be ratified). To prove the omni-band portion, it is necessary to demonstrate that the reconfigurable terminal works with bands other than EHF and that the terminal supports other types of waveforms. At this point, most of the research has been completed so the next logical step is to research the high data rate (HDR) waveform (this standard is also still in development). This step can be broken into the following items:

a. Complete the Universal Modem personality. This will require additional RF hardware and different processing.

b. Plan HDR research. This is the logical follow-on to the MDR work and the reconfigurable omni-band terminal work.

7.7 Step 6 - Industrial Development

Industrial participation will be solicited at all phases of the project. This should be done in parallel to the development. The industrial strategy is to cooperatively develop ground terminal technologies and to transfer this technology to industry to support possible Canadian Forces terminal acquisitions.

8. Conclusions

The concept of an omni-band reconfigurable terminal has been presented along with the design necessary for its construction. A stepping stone on the way is to build an MDR system simulator (which is later expanded to the omni-band reconfigurable terminal). This simulator allows DREO and CRC to develop the expertise in the field of MDR.

The MDR system consists of two main simulators: the payload simulator and the GT simulator. Both will be operated independently, so there is no overall system controller. The simulators will be connected by waveguide or coaxial cable to pass the RF signal between them. Each of the simulators will use similar hardware and will be distinguished by the personalities loaded in on power-up.

The simulators consist of a host processor, digital signal processing elements and the analog interface. The host processor will handle the user interface, simulator control, event logging, configuration control and download control. The digital signal processing elements will consist of GT or payload controller, access or resource controller, synchronization controller and communications control. The analog interface to RF is still being researched and will be specified in more detail in subsequent documents.

Within the simulators, the concentration is on capability to reconfigure using digital signal processors and if necessary reconfigurable logic boards. This will allow easy implementation of different standards for different communications systems. The digital signal processor chosen for the design is a powerful state-of-the-art chip from Texas Instruments. Reconfigurable logic boards will also be used to improve real-time performance. Personality information will reside in text files wherever possible allowing modification and customization using a normal text editor.

Debugging capability will be incorporated at the beginning of the design phase. This will facilitate development and simplify testing. It will also ensure a more reliable system as problems will be found early in development. This debugging capability includes built-in test circuits, self tests, sanity checks, bounds checking, as well as the reporting and response mechanisms. The user interface will be able to control the level of debugging and the resultant actions.

The sequence of development is described from the concept of an MDR simulator to industrial development of an omni-band reconfigurable terminal. The first two steps concentrate on baseband processing because the technology is currently more advanced in this area. Baseband processing involves the data communications, synchronization and access control portions. Meanwhile, research is progressing on the radio frequency portion. By the third step, the MDR simulator will be complete and capable of international interoperability testing. The next steps add low data rate (LDR) and Universal Modem (UM) personalities that allow further interoperability testing. Throughout the development, industry will be canvassed for involvement. The amount and timing of the involvement will depend upon the form of the cooperative venture. The industrial strategy is to cooperatively develop ground terminal technologies and to transfer this technology to industry to support possible Canadian Forces terminal acquisitions.

References

[1] *Satellite Data Link Standard for EHF Medium Data Rate Uplinks and Downlinks*, MIL-STD-188-136, Department of Defense, USA, 26 August 1995.

[2] *Military Standard Extremely High Frequency (EHF) Low Data Rate (LDR) Satellite Data Link Standards (SDLS): Uplinks and downlinks*, MIL-STD-1582C, Department of Defense, USA, 10 December 1991.

[3] *Simulation of MIL-STD-1582C TRANSEC Functions*, Project Report SC-96, Massachusetts Institute of Technology, Lincoln Laboratory, Lexington, Massachusetts, USA, 29 October 1993.

[4] R.D. Addison, "MDR/Omni-band Reconfigurable Terminal: Data Packet Specification", DREO Technical Note, September 1998.

[5] *TMS320C62x/C67x CPU and Instruction Set Reference Guide*, Texas Instruments Inc., March 1998.

UNCLASSIFIED

SECURITY CLASSIFICATION OF FORM
(highest classification of Title, Abstract, Keywords)

DOCUMENT CONTROL DATA

(Security classification of title, body of abstract and indexing annotation must be entered when the overall document is classified)

1. ORIGINATOR (the name and address of the organization preparing the document. Organizations for whom the document was prepared, e.g. Establishment sponsoring a contractor's report, or tasking agency, are entered in section 8.)

Defence Research Establishment Ottawa
Ottawa, Ontario
K1A 0Z4

2. SECURITY CLASSIFICATION (overall security classification of the document including special warning terms if applicable)

UNCLASSIFIED

3. TITLE (the complete document title as indicated on the title page. Its classification should be indicated by the appropriate abbreviation (S,C or U) in parentheses after the title.)

MDR/Omni-band Reconfigurable Terminal: Design Concept (U)

4. AUTHORS (Last name, first name, middle initial)

Addison, Robin D.

5. DATE OF PUBLICATION (month and year of publication of document)

September 1998

6a. NO. OF PAGES (total containing information. Include Annexes, Appendices, etc.)

46

6b. NO. OF REFS (total cited in document)

5

7. DESCRIPTIVE NOTES (the category of the document, e.g. technical report, technical note or memorandum. If appropriate, enter the type of report, e.g. interim, progress, summary, annual or final. Give the inclusive dates when a specific reporting period is covered.

DREO Technical Note

8. SPONSORING ACTIVITY (the name of the department project office or laboratory sponsoring the research and development. Include the address.

SST, Defence Research Establishment Ottawa
Ottawa, Ontario, K1A 0Z4

9a. PROJECT OR GRANT NO. (if appropriate, the applicable research and development project or grant number under which the document was written. Please specify whether project or grant)

5ca11

9b. CONTRACT NO. (if appropriate, the applicable number under which the document was written)

10a. ORIGINATOR'S DOCUMENT NUMBER (the official document number by which the document is identified by the originating activity. This number must be unique to this document.)

DREO TECHNICAL NOTE 98-007

10b. OTHER DOCUMENT NOS. (Any other numbers which may be assigned this document either by the originator or by the sponsor)

11. DOCUMENT AVAILABILITY (any limitations on further dissemination of the document, other than those imposed by security classification)

(X) Unlimited distribution
( ) Distribution limited to defence departments and defence contractors; further distribution only as approved
( ) Distribution limited to defence departments and Canadian defence contractors; further distribution only as approved
( ) Distribution limited to government departments and agencies; further distribution only as approved
( ) Distribution limited to defence departments; further distribution only as approved
( ) Other (please specify):

12. DOCUMENT ANNOUNCEMENT (any limitation to the bibliographic announcement of this document. This will normally correspond to the Document Availability (11). however, where further distribution (beyond the audience specified in 11) is possible, a wider announcement audience may be selected.)

Unlimited Announcement

13. **ABSTRACT** (a brief and factual summary of the document. It may also appear elsewhere in the body of the document itself. It is highly desirable that the abstract of classified documents be unclassified. Each paragraph of the abstract shall begin with an indication of the security classification of the information in the paragraph (unless the document itself is unclassified) represented as (S), (C), or (U). It is not necessary to include here abstracts in both official languages unless the text is bilingual).

In 1997, Defence Research Establishment Ottawa and Communications Research Centre started research into an omni-band reconfigurable terminal. Such a terminal will enable armed forces personnel to use a single ground terminal to communicate over any satellite communications or terrestrial link. Each ground terminal will support multiple standards. The first one to be implemented will be extremely high frequency medium data rate communications. There are two simulators: the payload simulator and the ground terminal simulator. Each simulator has a personal computer based host to run the simulator, a digital chassis containing several digital signal processor boards to run the payload or ground terminal, and a radio frequency chassis to interface to the digital boards.

Within the simulators, the concentration is on the capability to reconfigure. This will allow easy implementation of different standards for different communications systems. Debugging capabilities will be incorporated at the beginning of the design phase to facilitate development and simplify testing.

The sequence of development, from the initial concept of a medium data rate simulator to industrial development of an omni-band reconfigurable terminal, is given. Initial development concentrates on baseband processing with the radio frequency research in parallel. Midway through, the simulator will be ready for international interoperability testing. Later, additional standards will be supported. Throughout the development, industry will be consulted for the eventual transfer of the technology to industry for development of omni-band reconfigurable terminals for the Canadian Forces.

14. **KEYWORDS, DESCRIPTORS or IDENTIFIERS** (technically meaningful terms or short phrases that characterize a document and could be helpful in cataloguing the document. They should be selected so that no security classification is required. Identifiers, such as equipment model designation, trade name, military project code name, geographic location may also be included. If possible keywords should be selected from a published thesaurus. e.g. Thesaurus of Engineering and Scientific Terms (TEST) and that thesaurus-identified. If it is not possible to select indexing terms which are Unclassified, the classification of each should be indicated as with the title.)

satellite communications
MDR
omni-band
ground terminal
design
concept